Distance-vector routing protocol

From Wikipedia, the free encyclopedia

In computer communication theory relating to packet-switched networks, a **distance-vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman-Ford algorithm, Ford–Fulkerson algorithm, or DUAL to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically and, in some cases, when a change is detected in the topology of a network. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. *[citation needed]*

Distance Vector means that Routers are advertised as vector of distance and direction. 'Direction' is represented by next hop address and exit interface, whereas 'Distance' uses metrics such as hop count.

Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which or interface to which a packet should be forwarded.
2. Distance from its destination.

Examples of distance-vector routing protocols include Routing Information Protocol Version 1 & 2, RIPv1 and RIPv2 and IGRP. EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

Contents

Method

The methods used to calculate the best path for a network are different between different routing protocols but the fundamental features of distance-vector algorithms are the same across all DV based protocols.

Distance Vector means that Routers are advertised as vector of distance and Direction. Direction is simply next hop address and exit interface and Distance means hop count.

Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination.

As the name suggests the DV protocol is based on calculating the direction and distance to any link in a network. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as 'routing by rumor' because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

Limitations

Count-to-infinity problem

The Bellman-Ford algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like as A-B-C-D-E-F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down - B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down - so it tells B that A is only two jumps from C (C to B to A) , which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the "Relax property" of Bellman Ford).

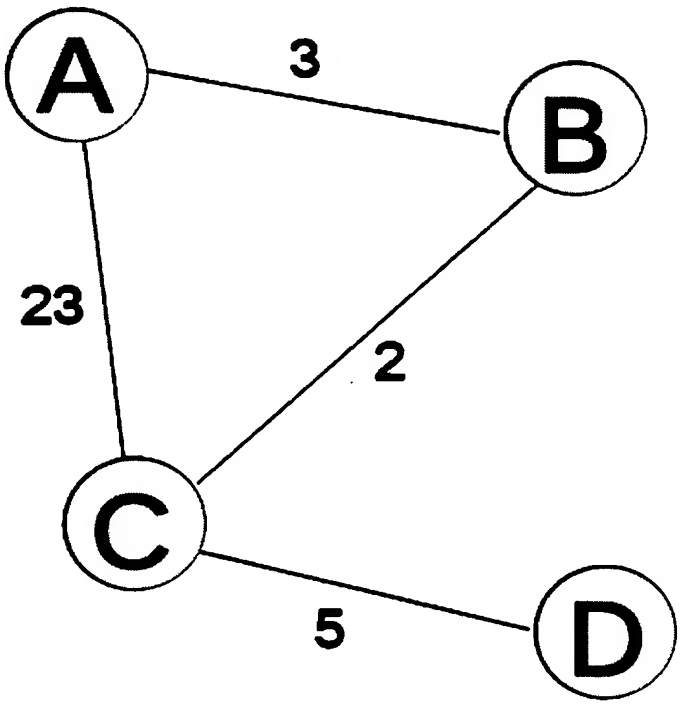
Partial solutions

RIP uses Split Horizon with Poison Reverse technique to reduce the chance of forming loops and uses a maximum number of hops to counter the 'count-to-infinity' problem. These measures avoid the formation of routing loops in some, but not all, cases. The addition of a *hold time* (refusing route updates for a few minutes after a route retraction) avoids loop formation in virtually all cases, but causes a significant increase in convergence times.

A number of loop-free distance vector protocols, such as EIGRP, DSDV and Babel, have been developed. These avoid loop formation in all cases, but suffer from increased complexity, and their deployment has been slowed down by the success of link-state routing protocols such as OSPF.

Example

In this network we have 4 routers A, B, C, and D:



We shall mark the current time (or iteration) in the algorithm with T, and shall begin (at time 0, or T=0) by creating distance matrices for each router to its immediate neighbors. As we build the routing tables below, the shortest path is highlighted with the color green, a new shortest path is highlighted with the color yellow.

T=0

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

from D	via A	via B	via C
to A			
to B			
to C			5
to D			

At this point, all the routers (A,B,C,D) have new "shortest-paths" for their DV (the list of distances that a them to another router via a neighbor). They each broadcast this new DV to all their neighbors: A to B a

to C and A, C to A, B, and D, and D to C. As each of these neighbors receives this information, they nov recalculate the shortest path using it.

For example: A receives a DV from C that tells A there is a path via C to D, with a distance (or cost) of 5 the current "shortest-path" to C is 23, then A knows it has a path to D that costs 23+5=28. As there are n shorter paths that A knows about, it puts this as its current estimate for the shortest-path from itself (A) t C.

T=1

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C
to A			28
to B			7
to C			5
to D			

Again, all the routers have gained in the last iteration (at T=1) new "shortest-paths", so they all broadcas DVs to their neighbors; This prompts each neighbor to re-calculate their shortest distances again.

For instance: A receives a DV from B that tells A there is a path via B to D, with a distance (or cost) of 7 the current "shortest-path" to B is 3, then A knows it has a path to D that costs 7+3=10. This path to D o 10 (via B) is shorter than the existing "shortest-path" to D of length 28 (via C), so it becomes the new "s path" to D.

T=2

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		33
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C
to A			10
to B			7
to C			5
to D			

This time, only routers A and D have new shortest-paths for their DVs. So they broadcast their new DVs neighbors: A broadcasts to B and C, and D broadcasts to C. This causes each of the neighbors receiving DVs to re-calculate their shortest paths. However, since the information from the DVs doesn't yield any paths than they already have in their routing tables, then there are no changes to the routing tables.

T=3

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C
to A			10
to B			7
to C			5
to D			

None of the routers have any new shortest-paths to broadcast. Therefore, none of the routers *receive* any information that might change their routing tables. So the algorithm comes to a stop.

External links

- A Java applet implementing Distance Vector algorithm for pedagogical purposes (http://www.mathiaz.com/routage/)

References and Further Reading

- "RFC1058 - Routing Information Protocol", C. Hedrick, Internet Engineering Task Force, June 1988 (http://www.ietf.org/rfc/rfc1058.txt)
- "RFC1723 - RIP Version 2 Carrying Additional Information", G. Malkin, Internet Engineering Task Force, November, 1994 (http://www.ietf.org/rfc/rfc1723.txt)
- "RFC2453 - RIP Version 2", G. Malkin, Internet Engineering Task Force, November, 1998 (http://www.ietf.org/rfc/rfc2453.txt)

- "A Path-Finding Algorithm for Loop-Free Routing, *J.J. Garcia-Luna-Aceves and S. Murthy, IEEE/ACM Transactions on Networking, February 1997*
- "Detection of Invalid Routing Announcements in the RIP Protocol", D. Pei, D. Massey, and L. Zhang, , IEEE Global Communications Conference (Globecom), December, 2003

Further reading

- Section "Link-State Versus Distance Vector" (http://docwiki.cisco.com/wiki/Routing_Basics#Link-State_Versus_Distance_Vector) in the Chapter "Routing Basics" in the Cisco "Internetworking Technology Handbook"

Retrieved from "http://en.wikipedia.org/wiki/Distance-vector_routing_protocol"
Categories: Routing protocols | Routing algorithms